Package ‘rank’

December 1, 2024

Title Customisable Ranking of Numerical and Categorical Data

Version 0.1.1

Description Provides a flexible alternative to the built-in rank() function called smartrank().
Optionally rank categorical variables by frequency (instead of in alphabetical order), and control whether ranking is based on descending/ascending order.
smartrank() is suitable for both numerical and categorical data.

License MIT + file LICENSE

Suggests covr, dplyr, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 2

Encoding UTF-8

RoxygenNote 7.3.2

URL `https://github.com/selkamand/rank`,
`https://selkamand.github.io/rank/`

BugReports `https://github.com/selkamand/rank/issues`

VignetteBuilder knitr

NeedsCompilation no

Author Sam El-Kamand [aut, cre, cph] (<`https://orcid.org/0000-0003-2270-8088`>)

Maintainer Sam El-Kamand <`sam.elkamand@gmail.com`>

Repository CRAN

Date/Publication 2024-12-01 22:30:02 UTC

Contents

smartrank — *Rank a vector based on either alphabetical or frequency order*

Description

This function acts as a drop-in replacement for the base `rank()` function with the added option to:

1. Rank categorical factors based on frequency instead of alphabetically
2. Rank in descending or ascending order

Usage

```
smartrank(
  x,
  sort_by = c("alphabetical", "frequency"),
  desc = FALSE,
  ties.method = "average",
  na.last = TRUE,
  verbose = TRUE
)
```

Arguments

`x`	A numeric, character, or factor vector
`sort_by`	Sort ranking either by "alphabetical" or "frequency" . Default is "alphabetical"
`desc`	A logical indicating whether the ranking should be in descending (TRUE) or ascending (FALSE) order. When input is numeric, ranking is always based on numeric order.
`ties.method`	a character string specifying how ties are treated, see 'Details'; can be abbreviated.
`na.last`	a logical or character string controlling the treatment of `NA`s. If `TRUE`, missing values in the data are put last; if `FALSE`, they are put first; if `NA`, they are removed; if `"keep"` they are kept with rank `NA`.
`verbose`	verbose (flag)

Details

If `x` includes 'ties' (equal values), the `ties.method` argument determines how the rank value is decided. Must be one of:

- **average**: replaces integer ranks of tied values with their average (default)
- **first**: first-occurring value is assumed to be the lower rank (closer to one)
- **last**: last-occurring value is assumed to be the lower rank (closer to one)
- **max** or **min**: integer ranks of tied values are replaced with their maximum and minimum respectively (latter is typical in sports-ranking)

- **random** which of the tied values are higher / lower rank is randomly decided.

NA values are never considered to be equal: for na.last = TRUE and na.last = FALSE they are given distinct ranks in the order in which they occur in x.

Value

The ranked vector

Note

When `sort_by = "frequency"`, ties based on frequency are broken by alphabetical order of the terms

When `sort_by = "frequency"` and input is character, ties.method is ignored. each distinct element level gets its own rank, and each rank is 1 unit away from the next element, irrespective of how many duplicates

Examples

```
# ------------------
## CATEGORICAL INPUT
# ------------------

fruits <- c("Apple", "Orange", "Apple", "Pear", "Orange")

# rank alphabetically
smartrank(fruits)
#> [1] 1.5 3.5 1.5 5.0 3.5

# rank based on frequency
smartrank(fruits, sort_by = "frequency")
#> [1] 2.5 4.5 2.5 1.0 4.5

# rank based on descending order of frequency
smartrank(fruits, sort_by = "frequency", desc = TRUE)
#> [1] 1.5 3.5 1.5 5.0 3.5

# sort fruits vector based on rank
ranks <- smartrank(fruits,sort_by = "frequency", desc = TRUE)
fruits[order(ranks)]
#> [1] "Apple"  "Apple"  "Orange" "Orange" "Pear"


# ------------------
## NUMERICAL INPUT
# ------------------

# rank numerically
smartrank(c(1, 3, 2))
#> [1] 1 3 2

# rank numerically based on descending order
```

```
smartrank(c(1, 3, 2), desc = TRUE)
#> [1] 3 1 2

# always rank numeric vectors based on values, irrespective of sort_by
smartrank(c(1, 3, 2), sort_by = "frequency")
#> smartrank: Sorting a non-categorical variable. Ignoring `sort_by` and sorting numerically
#> [1] 1 3 2
```

Index